IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

First Named Inventor:	David L. Donoho
Serial No.:	09/782,011
Filed:	February 12, 2001
Art Unit:	2445
Confirmation Number:	2182
Examiner:	Azizul Q. Choudhury
Title:	INSPECTOR FOR COMPUTED RELEVANCE MESSAGING
Attorney Docket No.:	UNIV0001 D2-C

April 6, 2009

Mail Stop Amendment
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

DECLARATION UNDER 37 C.F.R. 1.131(b)

This Declaration is submitted by the undersigned Dennis Goodrow who make the following declaration:

1. My name is Dennis Goodrow. I am over eighteen years of age and I am otherwise competent to make this declaration. I have first hand knowledge of the facts pertaining to the invention as set forth in this Declaration.

2. I am the Chief Architect of Big Fix, Inc., located in Emeryville, California.

3. Big Fix, Inc. is the Assignee the above identified U.S. Application No. 11/698,611.

4. Lisa Lippincott is an inventor for the above identified U.S. Application No. 11/698,611.

5. Lisa Lippincott is not available to produce an affidavit.

6. Beginning as early as September 17, 1997 and at least until September 1, 1998, I was an employee of Universe Communications, Inc., located in Berkeley, California.

7. Lisa Lippincott was an employee of Universe Communications, Inc. as early as September 17, 1997 and at least until September 1, 1998.

8. All of the subject matter claimed in the above identified U.S. Application No. 11/698,611, as of the date of this declaration, was invented, at least in part, by Lisa Lippincott.

9. The team of Lisa Lippincott, David Hindawi, and David Donoho (hereinafter "the Team") thought of the information discussed in the patent application as early as September 17, 1997. This date has been memorialized in the attached memorandum labeled as "Exhibit A", which was only available internally to select employees of Universe Communications, Inc. and was subject to a strict confidentiality notice.

10. Between September 17, 1997 and November 17, 1997 the Team was actively engaged in preparing designs and documents intended for disclosure to computer programmers employed by Universe Communications, Inc., such that they could create a working embodiment of the Inspector for Relevance Messaging.

11. On or around November 17, 1997, Mr. Scott Anderson, a computer programmer employed by Universe Communications, Inc., prepared the attached flow charts, labeled as "Exhibit B". Exhibit B shows, in part, workflow diagrams of the system of inspecting computer properties with relevance messaging. Exhibit B was available only internally to select employees of Universe Communications, Inc., and was subject to a strict confidentiality notice.

12. Between January 1, 1998 and June 9, 1998 Mr. Scott Anderson and I were involved in implementing the invention in a software version. The various stages of this implementation are chronicled in the attached charts, labeled as "Exhibit C." Exhibit C was available only internally to select employees of Universe Communications, Inc.

13. On or around June 2, 1998, the Team collaborated to prepare a technical document covering the essentials of creating a relevance messaging system. A version of this document, entitled "Creating an Advice Site, is attached herein, labeled as "Exhibit D." Exhibit D was available only internally to select employees of Universe Communications, Inc., and was subject to a strict confidentiality notice.

14. On or around June 10, 1998, the Team collaborated to prepare a disclosure document for the purpose of filing a patent application. A version of this document is attached herein, labeled as "Exhibit E." Exhibit E was available only internally to select employees of Universe Communications, Inc., and was subject to a strict confidentiality notice.

15. These statements are made with knowledge that willful false statements are punishable by fine and/or imprisonment under 18 USC 1001, and that any such willful false statement may jeopardize the validity of this application or any patent resulting therefrom.

D Goodrow

Date: April 3, 2009

Dennis Goodrow

Exhibit A

The AdviceNet RELEVANCE Language

Universe Communications, Inc.
2180 Dwight Way, Suite C
Berkeley CA 94704

9/17/97

Abstract

The RELEVANCE Language is a key component of the ADVICENET system, enabling authors of advisories to specify with precision the computers for which a certain advisory is relevant. This document describes the formal structure of the language and places in context the many considerations which have led to the current structure.

Contents

1 Preliminaries

This document describes the RELEVANCE language of ADVICENET . It aims only to provide basic information about the structure and capabilities of the language. It represents version α .07 and was authored on 9/17/97 .

A full appreciation of the language's use requires an understanding of the ADVICENET system, and can be obtained by careful study of the following documents.

- THE ADVICENET SYSTEM. Describes the current crisis in computer reliability and maintenance. Describes the ADVICENET system for communicating active advisories to computers worldwide. Describes the impact of this system on components of the existing crisis. Describes the development effort needed for a computer software/hardware manufacturer or a corporate intranet adminsitrator to benefit from ADVICENET .

- THE ADVICENET SITE DEVELOPER'S MANUAL. Describes the components of an advice site and how an advice provider can construct those components. Illustrates how an advice site can save an advice provider money in technical support and maintenance costs.

- THE ADVICENET INSPECTOR API. Describes how an advice provider can extend the RELEVANCE language to include capabilities directly addressing specific needs of the advice provider. Gives detailed information on the Applications Programming Interface and on the programming environment which is required to develop Inspectors.

- THE ADVICENET MAC OS INSPECTOR LIBRARY. Describes a layer of Macintosh-specific Types, Properties, Elements, and Casts which supplements the base RELEVANCE language with a comprehensive range of services for inquiring about the state of a MacOS computer.

- THE ADVICENET WIN95 INSPECTOR LIBRARY. Describes a layer of Windows '95-specific Types, Properties, Elements, and Casts which supplements the base RELEVANCE language with a comprehensive range of services for inquiring about the state of a Wintel '95 computer.

These documents are currently still under development.

2 Role of the RELEVANCE Language in AdviceNet

To set the stage for a detailed description of the RELEVANCE language, we begin with a sketch of the role played by the language in ADVICENET .

The ADVICENET system is based on the idea of a decentralized community of advice providers offering *advice sites* distributed worldwide throughout the Internet. These advice sites make available for HTTP access special documents called *advisories.* Advisories are ASCII text files resembling very much in format existing e-mail messages; in fact they are custom extensions of the e-mail/MIME message formats offered through Internet Standard RFC 822. It is expected that at some future date, the ADVICENET extensions will have proven so widely useful that they will become Internet standards themselves.

Advisories have a format resembling the format of e-mail messages, with many of the same components in the message/digest headers. The key extension offered by advisories is the institution of a new clause in the message: the *relevance clause*. The relevance clause is distinguished by the keyword `X-Relevant-When:`; what follows the keyword is an *expression* from the RELEVANCE language. It is this language that is being described in this technical report.

The purpose of this language is to enable the ADVICENET *Advice Reader*, running on a user's computer to *automatically* read an advisory and determine, *without intervention from the user*, whether the advisory is *relevant* to the user. Under the mediation of various user interface parameters set in the Advice Reader, the user will typically be notified of relevant advisories, and will be given the opportunity to *follow-through* as the advisories suggest. The actual follow-through action will depend on the advisory, but typically this will be an option to obtain and execute an application which renders, in a fully-automatic way, a repair to an undesirable situation.

This scheme means that it is possible for an advice author to publish advisories at its web site that solve many of the most common technical support problems faced by clients of the author's organization, and be assured that the advisories come to the attention only of users of precisely the computers that are in need of the specific advice being offered. The value in this procedure is that it saves humans the time and trouble of precisely matching solutions to computers in need of those solutions.

With this background, the RELEVANCE language can be seen as fitting into a context of

(i) Widespread distribution via Internet;

(ii) Unattended automatic parsing and evaluation;

(iii) User notification of *relevant* advisories only.

3 Design Goals of AdviceNet

The RELEVANCE language has been very specifically designed with this application background and purpose in mind. The design as it stands today aims at combining the following characteristics.

- *English-Accessible*. A relevance clause is, in principle, something an individual user could read and comprehend, though few users will choose to do so in most cases. The syntax of the RELEVANCE language resembles the syntax of plain english, with key roles in the language played by clauses formed from articles like *of*, *as*, *whose*, *exists*, and so on.

- *Descriptive*. The purpose of a relevance clause is to examine the state of an individual computer, and see whether it meets various conditions which could combine to imply the relevance of a certain advisory. In short the purpose of the language is to describe rather than to manipulate.

- *Nonprocedural*. Unlike many languages used in connection with the operation and/or maintenance of computers, the RELEVANCE language is not procedural: it does not specify how to manipulate the contents of various fragments of memory. This is the flip side of being descriptive: there is no useful purpose that would be served by enabling traditional procedural services: loops, assignments, and conditionals. Such services are not made available.

- *Open-ended.* The purpose of the language is to precisely describe the state of a computer. This state can change as the user purchases new software and/or hardware, or indeed as new software/hardware objects are invented. Consequently it is not possible to limit in advance what are all the components of state that the language will give access to; and the language is designed to give authors the ability to extend the language to express concepts about system state that haven't yet been conceived of.

- *Object-Oriented.* The language is extensible using the discipline of object oriented programming (OOP). The OOP buzzword means, in our case, that we offer a *Strongly-Typed, Polymorphic* language. We follow the strong-typing philosophy in the extreme. We carefully define the collection of all objects one would like to query, and carefully specify the methods for making queries, and forbid combinations that cannot be formed using our controlled collection of methods. We aim to avoid the usual procedural approach offering powerful, flexible, general purpose tools which can be used in many unanticipated ways. In the usual cases where that approach is used, it is good, but in the setting we are studying, it would be *dangerous.* OOP methodology allows us to design a controlled environment enabling powerful queries of the system state while remaining relatively secure from misuse and abuse.

- *Security and Privacy Aware.* Because the language is designed to be used in conditions of unattended operation over the Internet, it is very important that the language be safe in various ways. This concern has overwhelmed all others in the design of the product; it is fair to say that each of the distinctive features of the RELEVANCE language has been chosen based on these important concerns. It is believed that the language itself best addresses the security and privacy concerns that Internet operation will demand.

4 Security Issues in AdviceNet

The Internet is not a secure medium; along with the many services which it provides come the risks of exposing one's machines to potentially erroneous or even malicious agents. In designing the ADVICENET system, with its aspects of decentralized authorship and unattended operation, it is crucially important for the acceptance and usefulness of the system to address Internet security concerns.

We begin with the obvious comment that there is no such thing as an "ironclad" guarantee of security. The RELEVANCE language has been designed with several security issues in mind; the design addresses these issues, though there remains the possibility of other concerns we have not thought of. Your mileage may vary.

Our main concern is that the evaluation of clauses in the language not have damaging or embarassing side-effects. In evaluating clauses of the language, ideally we would have these four properties:

1. *No significant Claim on system resources.* Evaluation of a relevance clause should not consume inordinate amounts of CPU time, CPU memory, or hard drive space.

2. *No permanent change in system state.* The evaluation of clauses cannot effect the system, for example through creating or destroying files.

3. *No violation of user privacy.* The evaluation of clauses cannot result in an advisory communicating private data about the user to the outside world.

4. *Robustness against misuse of the language.* Desiderata 1-3 are not only supposed to hold in evaluation of well-formed clauses authored by those with "friendly intent". The evaluation of *arbitrary* clauses, even those which are malformed or which are authored by agents with "hostile intent", should *never* be able to lead to damaging side-effects. For example, the evaluation should be safe despite subtle mis-uses of the language – improperly formed clauses, misapplied operators, or mismatching data types.

These different concerns have dictated various aspects of the language design. For example, the RELEVANCE language is nonprocedural; this responds to security concerns, as it places limits on the resources which evaluation can consume. At an elementary level, the RELEVANCE language does not allow for recursion, for infinite loops, for arbitrarily-sized arrays to be allocated; these limits allow explicit bounds to be known (in principle) about the resources which a relevance clause can consume. A famous problem in logic concerns the ability to predict whether a machine evaluating an expression in a procedural language will ever reach completion. This is the *Turing Halting Problem.* It is not in general possible to decide whether a program in a procedural language will ever terminate. The situation in our setting stands in stark contrast:

> *All* RELEVANCE *expressions are decidable: they must halt.*

The object-oriented nature of the language also responds to security concerns. As the language is strongly-typed, it is not possible to apply operators to data which they were not designed to support. This allows to avoid crashes and dangerous misreferences, and to safely terminate the evaluation of expressions that are poorly formed.

A second facet of the object-oriented design which may be viewed as a security feature is the fact that manipulations of data are obtained not in the *surface level* of the RELEVANCE language itself but rather in the deeper level of the member functions of data types. These can be expected to be programmed more carefully than would be customary in the programming of a script, and to pay closer attention to clever use of computational resources. Moreover, they provide services which are available only after they are explicitly installed by the user, who therefore has a chance to apply scrutiny to the reliability and authenticity of the provider.

In general, our goal is that the user of ADVICENET should face no more exposure to security risks than a user normally suffers through routine Internet activity. We believe that in many ways ADVICENET creates fewer risks than existing widely used technology such as Web Browsers, in the forms in which such technology is often used (e.g. with "Cookie Dropping" enabled).

5 Privacy Issues in AdviceNet

Modern PC's contain an enormous amount of information about their users, some of this highly sensitive. It is very important for the acceptance and usefulness of ADVICENET that we aim at the goal

> *Information on the machine stays on the machine.*

In accessing advice and subsequent evaluation of relevance clauses we aim at two key principles

1. *Subscriber Anonymity.* No one should have a systematic way to know who is subscribing to a given advice site; in the standard AdviceNet system, it is not consider acceptable behavior to require the names and addresses of subscribers.

2. *No Feedback.* No one should have a systematic way to know which advice is relevant on which machines. There is nothing in the evaluation mechanism which can provide information about the user's machine to any other party.

This is false.

Point 2 is particularly worth examining. The lack of feedback remains true *even if a malicious party has published libraries which communicate using the Internet.* Because the Relevance language has a strict object oriented definition, and all object references lead to function calls with literal constants, it is not possible for a relevance clause to evaluate a function with an argument that is the result of an expression. Therefore even security holes cannot lead to publication of private information.

We aim for no more exposure to privacy risk than a user normally suffers through routine Internet activity.

6 More On Security and Privacy

The main ingredient of good security is the user's common sense: he/she should deal only with trusted, responsible advice providers. As described in the document *The AdviceNet System*, there are various ingredients in the system which, supported by trust mechanisms, deliver reasonable expectations of security.

An important ingredient is the division of labor between Relevance language and the rest of the system. The Relevance language does not offer the ability to fix problems, only to identify them. Fixing problems can only be done upon user approval. This gives a layer of insulation and a chance for reflection. The user is free to not approve that certain tasks be done; the user is free to study the situation before heeding an advisory, and in particular to inquire whether the advice itself is still being offered by the trusted site that first provided it, and whether the advice really was provided by that site. By these mechanisms, it is possible to protect trusted sources of advice, avoiding contamination by outside advice, pretending to be from a trusted source.

A further very important component is the recursive nature of the advice system itself. It is possible for an author to publish advice disowning earlier advice from the same author, recommending that it be removed from one's system; it is also possible to publish advice recommending not to heed advice from conflicting or dubious sources.

In short, the AdviceNet system contains a variety of tools addressing security issues; it is not the responsibility of the Relevance language alone.

It must also be admitted that there *are* security and privacy risks in AdviceNet . However, these essentially do not involve the Relevance language.

The greatest *security* risks involve what happens when a user decides to follow through on a piece of advice. At that point, in principle, anything is possible; when the user says "OK" he/she is effectively opening the machine to the advice provider who is then in a position to perform surgery of whatever magnitude.

It is likewise true that user approval poses the greatest *privacy* risks. The application that runs in response to user approval has, in principle, the ability to communicate to the Advice Provider, and there are no real restrictions on what can be communicated. An Advice Provider *can* know if a certain action (update or modification) something was approved by a certain user at a certain time. This can reveal useful information to the provider, since the provider is learning that a complicated condition on the system state is true. Depending on the complexity of the condition, the information revealed may be substantial.

Although the security and privacy risks that would be posed by improper design of the RELEVANCE language are not serious compared to the risks posed by indiscriminate user approval of untrusted follow-through, it is important to pay careful attention to these risks. It can be anticipated that ADVICENET will parse and evaluate many tens of thousands of relevance clauses for every clause that is found to be relevant. Hence many orders of magnitude more work will be done without user intervention than with user intervention. Risks posed by the evaluation process, even if suffered only rarely, might potentially have a numerically far more important role than other risks posed by the system.

It is also important to point out that in special circumstances, one might *not* want the security and privacy features to be available. One can anticipate that in a secure intranet setting, barred from outside access by a firewall, it would be useful for intranet managers to be able to know which machines on a corporate network respond 'Relevant' to certain queries. In that event, it would be useful to disable some of the restrictions we have placed on the language in the design discussed here. Our design goal emphasizes security and privacy, but in certain versions of the final product other emphases will be possible.

One can also imagine that certain advice providers will want to know who is subscribing to advice because they are selling advice and need to make it available only to paying customers. This arrangement violates the stated privacy goals underlying this document. However, while we emphasize privacy goals, we will also enable applications in which such goals are secondary.

7 Properties of the RELEVANCE Language

We now describe important details about the language structure and function. Given the earlier discussion, certain of the information presented here will appear repetitive; it seems best to be absolutely explicit about many details at this stage, and this seems to require repetition.

7.1 Descriptive Language

The RELEVANCE language is used for querying the state of a computer, a highly complex arrangement of software, hardware, and data. It is a description language which describes state rather than a procedural language which describes actions. As the language is not used for traditional procedural tasks (e.g. sorting data, moving data) it is *profoundly* different from a traditional procedural language. There are

- no named variables
- no assignment statements
- no function calls, or at least no explicit function calls with variable arguments
- no loops or conditional execution

Nevertheless, the language is designed to be powerful, in that it is intended to be *highly expressive*; a few words in this language provide access to answers about the system state which would be impossible to obtain in traditional programming languages short of writing hundreds of lines of code and invoking many specialized functions in system libraries.

Certain tasks can be done either in a descriptive language or in a procedural language; the code in a RELEVANCE language will typically look very different than the code for the same task in a procedural language, and the way of productively thinking about how to accomplish the same task will be very different between the two languages. It will be important for some programmers to remind themselves from time to time the main reasons why a relevance clause *should* be very different from a program in a procedural language. Three good reasons are

- Because procedural langauges pose security & privacy concerns;
- Because descriptive languages can be more economical (in terms of code length) for the purposes of writing relevance clauses;
- To make the distinction in purpose starkly obvious to programmers.

7.2 Layered Language Definition

It must be understood that the RELEVANCE language is very open-ended, and that it is built up in layer upon layer of extensions. In this document we only give detailed information about the base 'built-in' layer; with a few hints about other layers.

It must also be understood that the layers interface via object-oriented programming techniques. Hence the capabilities of each additional layer are delivered in packages "plugging-in" new object-oriented deliverables. In the RELEVANCE language these are data types, data properties, data elements, data casts, and operators.

Hence, to understand a completely installed system is to understand the layers which have been installed, and to understand the object-oriented services that each layer provides.

We envision that in a typical installation, these layers will go as follows:

- *Base Layer.* Contains the basic mechanics of clause evaluation: the Lexer, and the Parser for the language syntax, along with a number of Basic Built-in types, properties, elements, casts, and operators. It is expected that the base layer will be the same on every platform carrying ADVICENET .
- *System-Specific Layer.* This consists of Plug-ins which give basic system information about a certain family of computers. For example, one might be able to get the system date and time, the size of various files, the contents of the PRAM, or the names of attached peripheral devices. It is expected that these will all be in common for a given computer OS family – Win '95, MacOS – independent of manufacturer. It is expected that these will be similar from one OS to another – the object oriented structures and queries being basically the same – but perhaps with naming difference (e.g. "Folder" on one OS might be "Directory" on another).
- *Vendor-Specific Layers.* This collection of potentially a large number of "extensions Layers" would typically be produced by third parties giving special access to the internals of their hardware and software products: one can think of potential authors ranging a span of products from hardware producers (of, say, Cable Modems) to

software producers (of say Photoshop & Plug-Ins); to service providers (e.g. America On-Line).

It is expected that 80% or more of the power of the RELEVANCE language will be provided through services offered outside the Base layer.

We do not at the present time have any thoroughly thought-through examples of Class Libraries outside the Base layer. Obviously this is an important next step, which we expect to describe in two documents:

- THE ADVICENET MACOS INSPECTOR LIBRARY. Describes a layer of Macintosh-specific Types, Properties, Elements, and Casts which supplements the base RELEVANCE language with a comprehensive range of services for inquiring about the state of a MacOS computer.
- THE ADVICENET WIN95 INSPECTOR LIBRARY. Describes a layer of Windows '95-specific Types, Properties, Elements, and Casts which supplements the base RELEVANCE language with a comprehensive range of services for inquiring about the state of a Wintel '95 computer.

7.3 Expected Conditions of Use

We anticipate that many advice providers will use the RELEVANCE language under the following conditions.

- A programmer will author advisories containing relevance clauses.
- Relevance clauses will typically use types, operators, elements, properties from existing inspector libraries to obtain their expressive power.
- Very occasionally, the programmer will discover that a key property (say of a hardware product) is not made available through existing libraries, and will author such a property plug-in.
- The programmer's advice will depend on the existence of various plug-ins for its well functioning – both widely available pre-existing class libraries and his own newly-developed plugins. He will have the responsibility to verify that libraries are correctly installed. It is necessary to write meta-advice to verify this.

Thus the programmer has these responsibilities:

- To know the base language and system-specific inspector library;
- To write the relevance clauses for his advice using those inspectors;
- (Occasionally) to extend the collection of existing types and properties;
- To write meta-advice verifying that the language is properly configured to correctly evaluate his advice.

8 The Object Model

While a computer language involves both syntax and a collection of semantics for manipulating data structures, we have chosen in this document to begin with the role of data structures and semantics, and later to discuss syntax.

As discussed earlier, the RELEVANCE language is object-oriented, which means in our case that

- The language is *strongly-typed*: every object has a specific type, and operators and properties may only be applied to objects of appropriate types. (In some non-object-oriented languages, it is possible to misuse functions, applying them to data for which they cannot work properly. This is not possible in the RELEVANCE language).

- The language is *polymorphic*: the meaning of an operator or property reference depends essetially on the type of the direct object. This is also called *overloading* of operators & properties: the interpretation of an operator or property depends heavily on the data to which it will be applied.

The language differs from some other OOP languages in that the concept of encapsulation is de-emphasized. There is no distinction between private, public, and protected data. Or, more precisely, all data are public.

In the current revision of this document, Version α .07 , the important object-oriented notion of *inheritance* will not be discussed; although it is expected to ultimately be included in the system. See Section 8.4 below.

8.1 Objects and Methods

An *object* in the RELEVANCE language may be thought of as a reference to a physical object, such as a *file* or *SCSI device.* It is implicitly a data structure, to which can be applied access methods made available in the RELEVANCE language.

Every object has a *type*, and each type has associated with it several *accessor methods.* The language has four base types, which are essential to the workings of the language interpreter, and many extension types which are essential to the performance of useful work.

The base types are

- *Integer.* A signed 32-bit integer.

- *String.* A variable length character string.

- *Boolean.* A single bit.

- *World.* An abstract type, representing the computer of which the language is running.

The RELEVANCE language accessor methods can be thought of as giving answers to structured queries about objects. They are strongly-typed: an accessor may only be applied to a type for which a definition of that accessor for that type has previously been registered.

Accessors themselves have various classifications:

- *Properties*: methods which inspect an object and return a value.

- *Elements*: methods which inspect an object and under the control of a single parameter, return a value. Often these are components of the object's structure and hence are called elements. *Element-by-iIndex* takes an integer; *Element-by-Name* takes a string.

- *Operators*: methods which inspect an object (unary operators) or a pair of objects (binary operators) and return a value. These are invoked syntactically by the usual arithmetic and relational operators as commonly encountered in high-level languages; but there is no requirement that actual semantics be familiar. The interpretation of `a + b` need not resemble the usual interpretation – if `a` and `b` are not of integer type.

- *Casts*: methods which inspect an object and return a new object with a different type or, in some cases, the same type but a different format.

The purpose of the type-method system is to make available powerful query and manipulation tools but only under strong-typing restrictions that inhibit misuse.

8.2 Under The Table

The RELEVANCE language object system is intrinsically connected with the object system provided by C++.

Each Type, Operator, Property, Element, and Cast defined in the language corresponds to a class, an operator, or a public member function in C++

This isn't really true.

Types are classes; all the others are nonmember functions

In fact, extensions to the base set of types are provided by writing appropriate C++ code; see the document THE ADVICENET INSPECTOR API for an explanation how this is done.

We can view the library of extension types as providing a rich library of C++ routines for querying the system state and for manipulating the answers that get returned.

We can therefore view the RELEVANCE Language as a tool for providing access to such a library while imposing strong-typing restrictions which inhibit abuses.

8.3 The Object Universe

The bulk of the functionality of the RELEVANCE language is provided by the extension types which have already been mentioned. These can be conveniently organized as a mathematical graph structure, in which *nodes* represent *types* and *arrows* represent *accessor methods* which return properties, elements, or casts of those types.

The structure is rooted at `World`, which has many properties. On the Macintosh, a few of these are:

Properties of World

- System Folder. Returns an object of type *Folder* associated with the system folder.

- Boot Drive. Returns an object of type *Volume* associated with the boot volume.

- Processor. Returns an object of type *CPU* associated with the computer.

- Printer. Returns an object of type *Printer* associated with the currently chosen printer.

- Monitor. Returns an object of type *Monitor*.

- IPAddress. Returns an object of type *IPAddress.*

We can list a few elements as follows:

Elements of World

- SCSI Bus *Integer.* Returns an object of type *SCSI Device* associated with the specified SCSI Bus slot.
- Serial Port *Integer.* Returns an object of type *Serial device* associated with the specified SCSI slot.

It should be clear that, in general, there is a close connection between methods and the types that they return. Often these even have the same name. Remember, though, that the method that has name 'Processor' obtains a property of *World*, not of *Processor.*

The methods applicable to *World* generate a series of types with their own methods; those methods in turn generate other types. For example, an object of type *Volume* has properties *Driver* and *File-Allocation-Blocksize* and *Capacity* and *FreeSpace*, while an object of type *CPU* has properties *Name*, *Speed*, *FPU*, *MMX*, and so forth.

Some of these objects are of atomic type: *MMX* when applied to an object of type *CPU* has a Boolean type; while *File-Allocation-Blocksize*, when applied to an object of type *Volume* has an integer type. However, some of these objects are of new type: witness *Driver*, which is an object with properties such as *Vendor* and *Version.*

Above the
table

`atomic type"
doesn't really
have a meaning.
Below the
table, no guard
types are atomic.

In a future version of this document we expect to work out in detail all the types and methods associated with two areas of special interest:

1. Internet Settings example
2. Volume examples

8.4 The Evolving Universe

The object model we are describing here is still under development. Two major enhancements to the system described here are currently under development.

- *Inheritance.* A basic platform-independent type (but with platform-dependent implementation) is *File.* This is an object with the following properties on MacOS:
    - Name. Object of type *Filename.*
    - Creation Time. Object of type *Time*
    - Modification Time. Object of type *Time*
    - Length. Object of type *Integer.*
    - Checksum. Object of type *Integer.*
    - Parent Folder. Objcct of type *Folder.*
    - Version. Object of type *Version.*
    - Type. Object of type *MacFileType.*
    - Creator. Object of type *MacFileCreator.*

Under MacOS, there are also files with special properties, and hence having their own specialized types: *Application*, *Control Panel*, *Extension*. For example, an application has all the properties of a file as well as a *Partition Size*, *Recommended Partition Size*, *Minimum Partition Size*, and *Bundle Bits*. A control panel has all the properties of file, as well as *enabled*.

In traditional OOP systems, one would handle such issues by creating a base type *File* and derived types *Application*, *Control Panel*, etc. The derived types accept all the same accessor methods as the base type *File*, and objects of type *Application*, say, accept also the accessor methods for that type, such as *Partition Size*. We expect that the RELEVANCE language will ultimately do this, though the details remain to be worked out.

- *Plural Properties*. It is important for certain purposes to know that there only be one of a certain object in existence, and for other purposes, it does not matter if there be one or more than one instance matching a reference.

  As a convenient way to make it possible to verify either circumstance, the RELEVANCE language will offer plural properties. For example, the property *application* is singular and the property *applications* is plural. The distinction is this

  - `application "Netscape"` asserts the existence of *exactly one* application of type netscape in the scope. It will fail if there is no such application and also if there are more than such application.
  - `applications "Netscape"` asserts the existence of *one or more than one* application named "Netscape" in the scope. It will fail only if there is no such application.

  I was planning to allow zero

  The existence of singular and plural properties means that phrases have (by inference) singular or plural characteristics. Accomodating this will require certain modifications to the grammar and semantics of the language, which remain to be worked out.

8.5 Describing the Universe

In order to keep track of all the types and methods available in ADVICENET , we will make available a *Type Inspector*, a specialized dictionary indicating all the available types and their associated methods and the results of thse methods.

This dictionary will be rendered in two different ways. In an on-line version, it will offer a hypertext-format dictionary describing each relevant term and the associated definition and relationships. This will be reminiscent, for some readers, of the Class Dictionary in AppleScript [1].

In a printed version, it will take a standard format reminiscent from typical system manuals for UNIX and related operating systems. Roughly speaking, we expect a single-page description of each type, taking the form

Typename – *1-line synopsis*

Description. - 1 paragraph of text describing the general role of this type, and important considerations for those making use of it.

Contained-in. – Name of library that installs it. URL of library source. Current version of that library..

Properties.

Prop1 Value-Type. Description of what the value is supposed to be, and the type that it takes.

...

Elements.

keyword1 name Value-Type. Description.

keyword2 integer Value-Type. Description.

...

Unary Operators.

unop1 Value-Type. description.

...

Casts.

as **Typename1** description

...

How-reached. *Description of Types from which this type is reachable as the value of a property or element.*

property1 of Type1

...

Related Types. *Description of Types which have intimate connections with the current one, either*

Type1 Reason.

...

Subtleties. *Hairy points to keep in mind.*

1. Note1.

...

9 The Evaluation Model

We now turn to rules for successful interpretation of clauses in the Relevance language.

9.1 Abort, Retry, Fail

The basic purpose of the Relevance language is to successfully lex, parse, and evaluate a single expression, resulting in a value of type *Boolean*. The resulting value is passed to the AdviceNet advice reader for further processing.

Of course, there will always be cases where a Relevance clause is malformed or otherwise unacceptable. In order to deal with that situation in an organized fashion, the lexer/parser/interpreter follows this rule.

> A Relevance Clause will lead to an advisory being declared relevant *only* if, the clause is *intelligible*, *successfully evaluates* to a result of type *boolean*, and is *true*.

Let's explain this rule in more detail:

- Expressions can be *Unintelligible* or *Intelligible*. *Unintelligible* expressions occur for one of three reasons.
  - *Lexical unintelligibility.* For example, the expression contains bizarre unquoted characters, such as $.
  - *Syntactic unintelligibility.* For example, the expression contains contains disallowed usage, such ++name.
  - *Run-time unintelligibility.* For example, the expression refers to an unknown property (e.g. `exists solarplexus of file`).
- Intelligible expressions can *Succeed* or *Fail* to evaluate.
  - *Arithmetic Failure.* 65536*65536 overflows type *integer*.
  - *Reference Failure.* `picture resource 101` doesn't exist.
  - *System Failure.* There is an error reading the hard drive.
- Successful evaluations can be *Boolean* or not. Examples of types which are not Boolean: *integer*, *IPAddress*, etc.

9.2 Exception Handling

The Relevance language interpreter creates two special objects to indicate evaluation failures.

- NSO, which stands for 'No Such Object'.
- TMO, which stands for 'Too Many Objects'.

For example the expression `length of file "foo"` evaluates to NSO if file `"foo"` doesn't exist.

NSO *generally* but not always causes expressions to FAIL to evaluate. There are at the moment two exceptions:

1. `exists(NSO)` evaluates successfully.
2. `exists folder whose ( length of file "foo" of it is 8 )` evaluates successfully if SOME folder contains a file `"foo"`, even though many folders do not.

The expression `application "Netscape"` evaluates to TMO if there are *two or more* applications which are instances of Netscape.

TMO *generally* but not always causes expressions to FAIL to evaluate. There are at the moment two exceptions, analogous to the exceptions for NSO discussed above.

10 Relevance Language Syntax

We finally attempt a description of the syntax of the Relevance language!

10.1 Simple Examples

1. Existence of a certain application.

```
X-Relevant-When: exists application "Photoshop"
```

2. Comparison of version numbers.

```
X-Relevant-When: ~~exists Control Panel "MacTCP" and~~
                 version of Control Panel "MacTCP" is ~~2.02~~ version "2.02"
```

3. Compare modification dates.

```
X-Relevant-when: ~~exists Photoshop PlugIn "Picture Enhancer" and~~
                 modification time of Photoshop PlugIn "Picture Enhancer" is
                 greater than time "~~10 January 1997~~"
                                   "10 Jan 1997 00:00 gmt"
```

4. Examine Control Panel Settings.

```
X-Relevant-When: exists Control Panel "ConfigPPP" and
                 Transport Mechanism of Control Panel "MacTCP" is equal to "SLIP"
```

We now turn to a more systematic description of the language; we consider in turn: Lexical analysis, Syntax, and Semantics.

10.2 Lexemes

The lexical analysis of the string breaks up the input string into tokens consisting of basic elementary inputs.

In the following, we will denote literal keywords by enclosing them in curly braces {`like this`}, and we will denote general Lexeme Classes by [class name].

Lexical tokens come in one of the following basic categories.

[String] A string of printable ascii characters enclosed in quotation marks (").

[Integer] A potentially signed string of decimal digits.

[Minus] The character -.

[SumOp] The characters +-.

[PrdOp] The characters */.

[RelOp] The character sequences = > >= <= !=.

[Phrase] A sequence of one or more unquoted words.

These categories are mostly unexceptional, and will be familiar to programmers from work with other computer languages.

10.2.1 Phrases

The most non-standard aspect of the language is in the definition of token type [Phrase]. Formally, a Phrase is a string of words; a word is a string of characters beginning with a letter. Any Phrase can be further decomposed into several occurrences of [Reserved Phrase], with, intervening word sequences (by definition) called [Ordinary Phrase].

Here is a partial list of
Reserved Phrases.

- as
- and
- containing
- ends with
- exists
- is
- it
- number of
- or
- remainder
- starts with
- whose

These phrases are all built into the base layer of the language, and have purposes associated with the base functions of the language; hence they are reserved from other use.

Here is a partial list of
Ordinary Phrases.

- Control Panel
- Photoshop PlugIn
- Modification Time
- MacFileType
- MacFileCreator
- SCSI Bus
- Shared Disk
- CD ROM

These are all phrases that describe system-dependent concepts, and so they are associated with the System-specific or Vendor-specific layers of the language. The collection of ordinary phrases is not fixed in advance, and can be exepected to grow with time as more inspectors are added to the language.

We note that one has *Run-Time Unintelligibility* when the lexer extracts an ordinary phrase from a string of words and cannot find that phrase in the current collection of installed libraries.

10.2.2 Special Notes

1. Strings are simply quoted sequences of ASCII characters.
   - There is no a-priori length limit on strings. It is a *System Evaluation Failure* if the lexer encounters strings so long that they cannot be stored in available RAM.
   - RFC 822 imposes length restrictions on ~~strings.~~ line length
   - There will be a mechanism for continuing strigs across lines.
2. Escape sequences will be specified, but have not yet been defined. Most likely usage is to specific ASCII codes explicitly by $\backslash D_1D_2D_3$ where the D_i are decimal digits, although hexadecimal and other coding schemes might be offered. However, issues of accents and other items that appear in connection with foreign character sets will have to be carefully studied.

10.3 Formal Grammar

In the table below, we denote concepts defined in the left-hand-side of Grammar Productions by ⟨name⟩

```
<Goal>          :=  <Expr>

   <Expr>          :=  <Expr> {or} <AndClause>   | <AndClause>

   <AndClause>     :=  <AndClause> {and} <Relation> | <Relation>

   <Relation>      :=  <SumClause> [RelOp] <SumClause> | <SumClause>

   <SumClause>     :=  <SumClause> [SumOp] <Product>
                   |   <SumClause> [Minus] <Product>
                   |   <Product>

   <Product>       :=  <Product>   [PrdOp] <Unary>
                   |   <Unary>

   <Unary>         :=  [Minus] <Unary>
                   |   [UnyOp] <Unary>
                   |   <Cast>

   <Cast>          :=  <Cast> {as} [Phrase]
                   |   <Reference>

   <Reference>     :=  [Phrase] {of} <Reference>
                   |   [Phrase] [string]    {of} <Reference>
                   |   [Phrase] [integer]   {of} <Reference>
                   |   [Phrase] <Restrict> {of} <Reference>
                   |   [Phrase] [string]
                   |   [Phrase] [integer]
                   |   [Phrase] <Restrict>
                   |   [Phrase]
                   |   {exists} <Reference>
                   |   {number of}  <Reference>
                   |   [string]
                   |   [integer]
                   |   {it}
                   |   ( <Expr> )

   <Restrict>      :=  {whose} ( <Expr> )
```

Semantic Notes.

11 To Probe Further

- Relation to AppleScript.
- Readings on OOP.

12 Appendix 1. Properties of World

13 Appendix 2. The Base Library of Types and Accessors

References

[1] The Tao of Apple Script.

[2] The Tao of Objects.

[3] THE ADVICENET SYSTEM.

[4] The AdviceNet Site Developer's Manual.

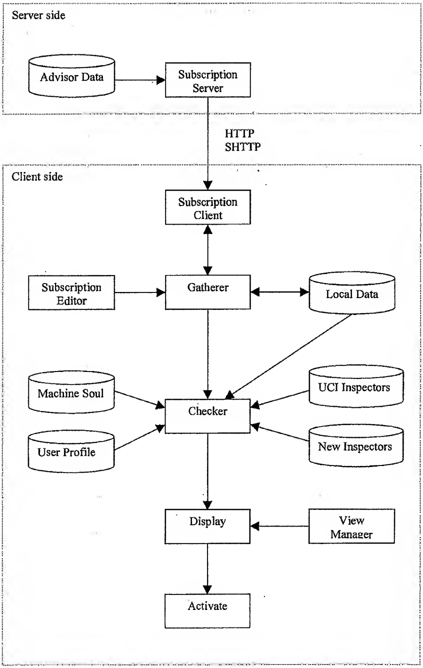
[5] The AdviceNet Inspector API.

[6] The AdviceNet Mac OS Inspector Library.

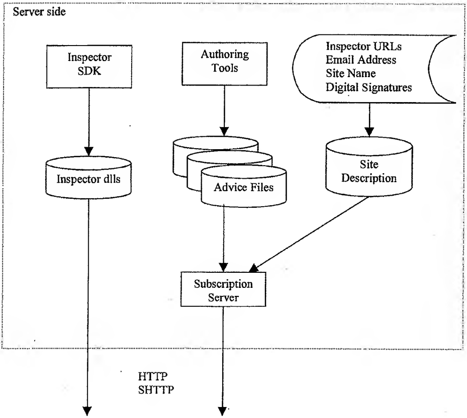
[7] The AdviceNet Win95 Inspector Library.

Exhibit B

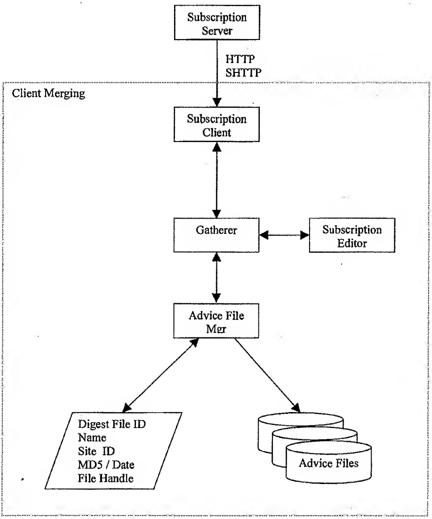
Overview


Server side
Advisor Data
Subscription Server
HTTP
SHTTP
Client side
Subscription Client
Subscription Editor
Gatherer
Local Data
Machine Soul
User Profile
Checker
UCI Inspectors
New Inspectors
Display
View Manager
Activate

Server Side Detail

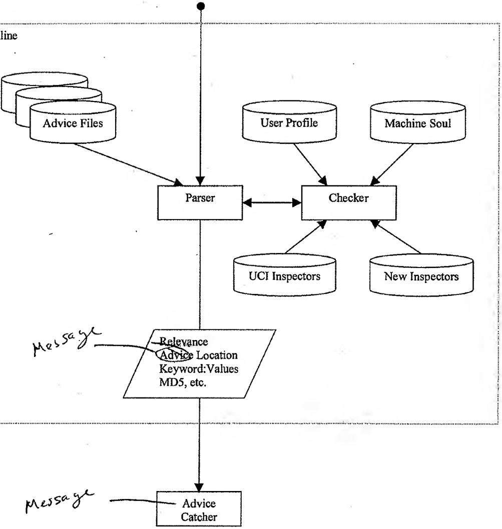


Client Data Gathering

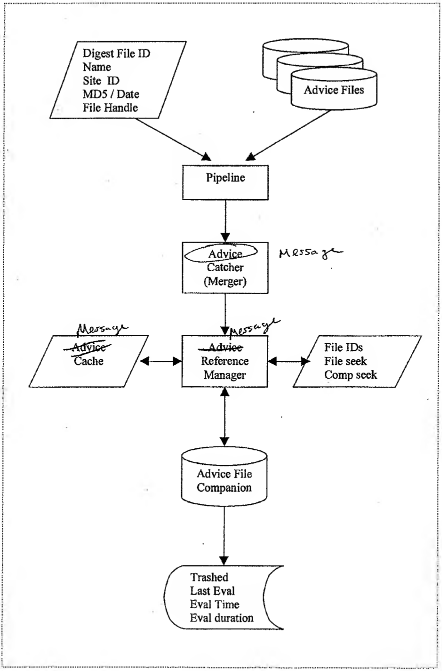


Delete?
Rename?

Pipeline


Pipeline
Advice Files
User Profile
Machine Soul
Parser
Checker
UCI Inspectors
New Inspectors
Message
Relevance
Advice Location
Keyword:Values
MD5, etc.
Message
Advice Catcher

Client Advice Evaluation



Subscription View

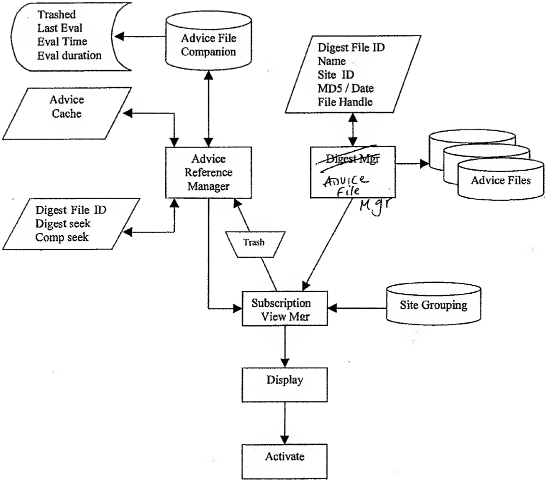
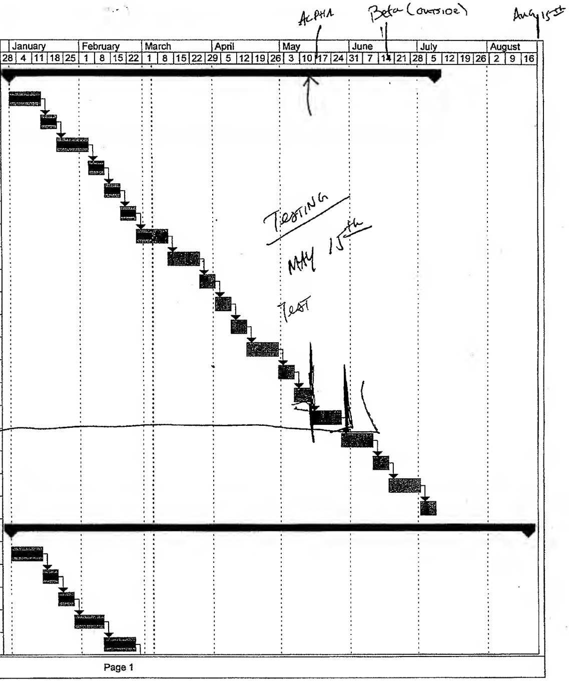


Exhibit C

ⓘ	Task Name	Duration	Start
	Dennis	**135 days**	**Thu 1/1/98**
✓	Advice File Manager	10 days	Thu 1/1/98
✓	Feed html display	5 days	Thu 1/15/98
✓	Companion Data	10 days	Thu 1/22/98
✓	Idle relevance evaluation	5 days	Thu 2/5/98
✓	Save / Load program state	5 days	Thu 2/12/98
✓	Site Synchronization	5 days	Thu 2/19/98
	Client Side File Transfer	10 days	Thu 2/26/98
	Scheduling & Gathering	10 days	Thu 3/12/98
	CGI-Bin Directory Reader	5 days	Thu 3/26/98
	Find	5 days	Thu 4/2/98
	minimized mode	5 days	Thu 4/9/98
	User Profile	10 days	Thu 4/16/98
	email	5 days	Thu 4/30/98
	Save html message	5 days	Thu 5/7/98
	Catching Errors	10 days	Thu 5/14/98
	Execute Vbscript, JavaScript	10 days	Thu 5/28/98
	Browser helper app	5 days	Thu 6/11/98
	Installation	10 days	Thu 6/18/98
	Inspectors/Authoring tools	5 days	Thu 7/2/98
	Scott	**164 days**	**Thu 1/1/98**
✓	html display	10 days	Thu 1/1/98
✓	Window management	5 days	Thu 1/15/98
✓	site statistics	5 days	Thu 1/22/98
✓	Subscription manager(add, dele	9 days	Thu 1/29/98
✓	View by Site	10 days	Wed 2/11/98



❶	Task Name	Duration	Start
✓	Load / Save UI state	5 days	Wed 2/25/98
	Start Patent search	5 days	Wed 3/4/98
	Inspectors/Authoring tools	5 days	Wed 3/11/98
	Document Internals	10 days	Wed 3/18/98
	Start Web site creation	25 days	Wed 4/1/98
	Sales Calls	5 days	Wed 5/6/98
	Preferences Proxy	5 days	Wed 5/13/98
	Help files	10 days	Wed 5/20/98
	Document User manual	25 days	Wed 6/3/98
	Web Marketing	30 days	Wed 7/8/98
	Jim	113 days	Thu 2/5/98
✓	Callback (list management)	10 days	Thu 2/5/98
✓	Owner Draw	5 days	Thu 2/19/98
✓	Window sizing / Progress Bars	5 days	Thu 2/26/98
	Scheduling & Gathering (UI)	5 days	Thu 3/5/98
	Sorting Columns	5 days	Thu 3/12/98
	Trash Interface	5 days	Thu 3/19/98
	Trash management	5 days	Thu 3/26/98
	Printing	5 days	Thu 4/2/98
	Inspectors/Authoring tools	15 days	Thu 4/9/98
	SDK for Inspector Writers	15 days	Thu 4/30/98
	Syntax Checker . Authoring tool	10 days	Thu 5/21/98
	Icons in list	3 days	Thu 6/4/98
	Inspectors/Authoring tools	25 days	Tue 6/9/98

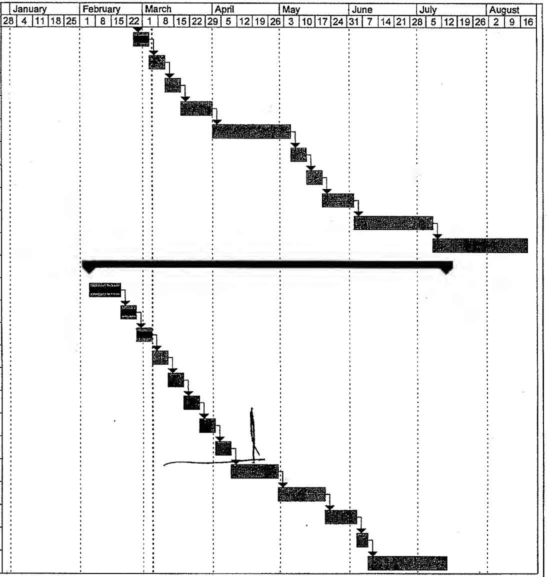


Exhibit D

Creating an Advice Site

A Guide to Creating an AdviceNet Site

Universe Communications, Inc.
2180 Dwight Way, Suite C
Berkeley, CA 94704

Last Modified: June 2, 1998

The information contained in this documentation is **Confidential** and **Proprietary**. It is not for distribution outside of **Universe Communications, Inc.**, which holds the Copyright on the document and all trademarking and patent rights covering names, concepts, and procedures contained inside.

Contents

1 Advisor Document Map

	The Big Picture	For end-users
	Using the Program	For end-users
	Visiting Our Site	For end-users
	Authoring Advice	For Advice Authors
	The Relevance Language	For Advice Authors
	The Win95 Inspector Library	For Advice Authors
	The Mac Inspector Library	For Advice Authors
	The Authoring Tools	For Advice Authors
You are here →	**Creating an Advice Site**	**For Advice Site Developers**
	Vendor Guidelines	For Advice Site Developers
	Technical Overview	For IT Managers
	Using the IT Manager Program	For IT Managers
	Writing Inspectors	For Inspector Writers

To get the most out of these documents, you should read them in the order indicated. For this document, you should have already read the three end-user manuals and all the pertinent advice authoring manuals.

You should have also downloaded the authoring tool programs from http://www.advisories.com.

?

2 Introduction

and supporting media files

This document covers the essentials of creating an ***Advice Site*** for distributing relevant advice to a targeted audience. An advice site is a collection of relevant advisories on a server that your customers can subscribe to. These people visit your site on a regular basis to synchronize with your latest advice. As with all AdviceNet advisories, your customers see only what is directly relevant to them.

Why would you do such a thing? Here are just a few sample scenarios:

- You're an **OEM computer maker**. You've developed an internal database to determine which intersection of programs, hardware and peripherals can cause problems. When someone calls in, your help desk guides them through a list of questions designed to identify the exact problem and deliver a solution. Each call costs you $20. When you create an advice site, you give out the answer once, not once per customer. Another plus: customers who need the advice will automatically get it -- *and no others*.
- You're a **software publisher**. You've got a hit on your hands, but of course you pushed the envelope a little, and video card vendors are scrambling to catch up with your latest technology. You find out that on some computers, the wrong video driver can render the computer almost unusable. You want to reach these people before they boot your program. Fortunately, you put the AdviceReader on their CD and gave your users a complimentary subscription to your advice site. When they receive their first advisory -- *before* running the game -- their computer is probed. If they have a problem driver, the AdviceReader will let them download the latest driver. The bug will never happen.
- You're a **hardware manufacturer**. You make an adapter for laptop computers. You find out that there can be a shock hazard when your adapter is used with certain laptop models. You create an advisory at your site that checks to see if the computer is one of the affected ones, and offer a free replacement. Only the customers who are in danger are warned -- all others are spared concern. This is a serious enough problem that you might also consider urgent advisories and postings to the AdviceNet main site.
- You're a **consultant**. You have a group of high-paying corporate customers that are using a suite of software you configured for them. As they report problems, you create advice and post it to servers at each customer's site. From there, the corporate employees can get access to your exclusive advisories. As a lone consultant, it gives you the clout of an entire help desk, capable of dealing with hundreds or thousands of customers.
- You're an **IT manager**. You lose a lot of sleep wondering about the latest bugs to eat their way through your software jungle. To help you catch a few winks, you set up a site in a shared directory on your company's network. Now when you find out about a problem, you can post a solution that every computer will see bright and early each morning -- or every hour if you want. In a well-characterized environment like this, you can come close to the ideal advisory that can reach *every* afflicted user and *none* of the others.

As you can see, there are some compelling reasons for creating and maintaining your own advice site. If you want to reduce tech-support calls while simultaneously improving public relations and the perceived value of your product, read on.

There are many ways to create advisories and send them to people. Since the files are MIME-compliant (see the *Authoring Advice* document), you can email them to customers and colleagues. If you're dealing with urgent advice, this can be a fine technique. But establishing a continued presence with a permanent site gives you more credibility, and credibility is a big part of the online success formula.

> Customers ... are guaranteed to come, since you're offering free advice that's always relevant and never intrusive.

Along with credibility is branding. A site can help you build a brand. You need logos, mottoes and jingles to really express and identify yourself. Since your advice can be composed in HTML, you can bring all the bells and whistles of your brand into the advice world. Although the online community is considered to be more democratic, it is also more distant. In a situation like this, where people can't just walk into your storefront and talk to you, branding is more important than ever. Treat your brand with respect -- it's how people get to know you online.

If you already have a branded presence, you'll appreciate the measures that the AdviceReader has taken to help you maintain that momentum. Logos, icons and bitmaps that you provide are spotted liberally throughout the interface to remind the user of who supplied the advice.

An advice site is the ideal complement to an existing marketing plan. It's a perfect way to keep in touch with your customers: they are guaranteed to come, since you're offering free advice that's always relevant and never intrusive. At this point, your advice site transcends its utility as a tech-support tool and becomes a major selling point.

Once you've got your users hooked, you can provide gentle reminders of updates and revisions. Thus a well-crafted site increases the value of your product, reduces the cost of supporting it and provides you with marketing leverage.

By the end of this document, you'll be ready to create your own advice site, and begin to offer your customers world-class customer support.

3 Picking Your Name

Before you do anything else, you should check the Advisor name registry to claim a name for your site. This registry is designed to prevent naming conflicts, especially with site profiles (which we'll return to later). You can still set up a site if you don't register, but you will limit the utility of your offerings and you may end up forcing your users to decide between two sites with the same name. It could be *your* site they decide to drop.

Registration also gives you greater credibility. It separates you from fly-by-night advisors who aren't courteous enough to take this basic, reassuring step. All the major vendors on AdviceNet are registered; it protects their brand and helps their subscribers to feel safe.

Additionally, the registry list is publicly available for users to consult when they want to subscribe to a new advice site. You could consider it a form of advertising.

To view the registry and reserve your name, go to http://www.advisories.com/namereg/.

4 Planning your site

AdviceNet is an extensive system that includes some powerful tools. There are many ways of setting up a site to achieve your exact goals. Don't be intimidated by the power of the system. If you've ever created a web site and you know how to FTP files around, you're half the way there. In

SUBDIRECTORIES are ignored.

general, a site must have at least one advice file in the root directory. Other than that, the world is open to all kinds of possibilities. There are several different types of files you may want to use, and these are all discussed in the next section (*Using the Files*).

How you put these ~~files together~~ determines the kind of site you will create. Here are the three basic site types:

4.1 Local Site

Networked file system

This is a *directory* you subscribe to, often on a network with shared access. A local site might be maintained by an IT director at a company that wants to centralize internal tech support.

4.2 Remote Site:

This is a typical installation on a *server that does not provide CGI* ~~*access*~~ services. As well as HTTP, it offers FTP access. It uses a static file to represent the directory structure. When a user visits this site, this static file will be consulted to see if anything has changed. It will work well for a single site, but you must remember to update the static file every time you change your advice, or it will never get downloaded.

4.3 Remote Site with CGI

This is the power method for installing *multiple sites on a server*. The CGI program automatically notices when the advice changes, so manual intervention isn't necessary when you update your advice. If your site gets a lot of traffic, you may want to consult with your server administrator about caching and other performance boosters.

All site types have several files in common, as well as certain files that are specific to its particular purpose. The following table illustrates the file structure of the three site types:

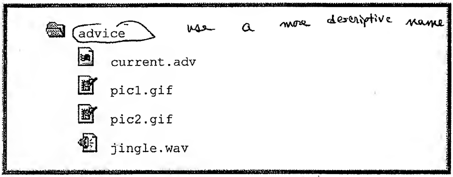
Site Type:	Local	Remote	Remote with CGI
Location:	Drive Path	URL	URL
Files:	Advice File (adv)	Advice File (adv)	Advice File (adv)
	Site Definition File (adf)	Site Definition File (adf)	Site Definition File (adf)
		URC filem (urc)	CGI Program (exe)
Optional:	Logo (gif, jpg)	Logo (gif, jpg)	Logo (gif, jpg)
	Graphics (gif, jpg)	Graphics (gif, jpg)	Graphics (gif, jpg)
	Sound (wav)	Sound (wav)	Sound (wav)
	Animation (ani, gif)	Animation (ani, gif)	Animation (ani, gif)

5 Using the Files

If your site is on a network as a shared directory, you just need to create your advice file and place it in the chosen directory. If you have any sound or graphics files that are referenced by HTML advice, put them in the same directory. The directory might look like this:

example

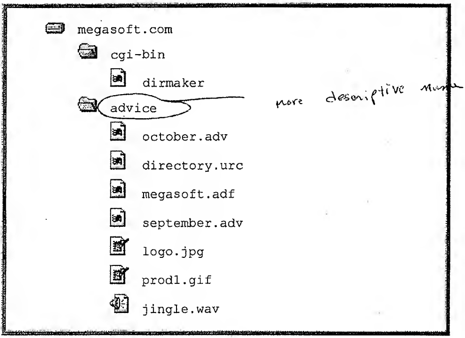
MegaSoft Printer/...



```
advice
    current.adv
    pic1.gif
    pic2.gif
    jingle.wav
```

use a more descriptive name

If you need a site that the whole world can access, you'll want to place your files on a server. Again, the most important file is the advice file itself. In addition, you'll need a URC file that contains a listing of the advice directory. After that, you can add any other auxiliary files (logos, graphics, sound files, etc.) you desire to make an attractive site. You may also have CGI programs. If so, they are usually in a separate directory. Here's a sample site layout for a server at MegaSoft:



```
megasoft.com
    cgi-bin
        dirmaker
    advice
        october.adv
        directory.urc
        megasoft.adf
        september.adv
        logo.jpg
        prod1.gif
        jingle.wav
```

more descriptive name

Exhibit E

1. Problem being addressed

Advice Provider:

- Organization or individual represented by server on an intranet or internet.
- Knows of conditions under which certain consumers would like to know something,
    - potentially to act on it with approval.
- Potentially thousands or millions of conditions it can offer advice about.
- Potentially millions or billions of individuals it can offer advice to
- Most conditions depend on very special combination of circumstances at consumer end
    - Affect only a small fraction of consumer base, but large number of consumers
    - Description of condition might involve knowing a great deal of detailed information
        - about the computer or contents of its storage devices
- This information might be considered very sensitive by consumers

Advice Consumer:

- Organization or individual represented by client computer on an intranet or extranet..
- Knows of Advice Providers offering advice of potential benefit to consumer.
- Typically does not want to review all the advice being offered by Advice Provider; only
    - wants to see the subset of advice which is relevant and timely.
- Typically does not want to reveal information about his identity or detailed condition of his
    - computer or contents of its storage devices to Advice Provider.
- Wants possibility of evaluating advice before adopting it.
- Wants possibility of adopting advice automatically.
- Can have relationship of this type with tens or thousands of Advice Providers.

AdviceNet System:

- Connecting AP with AC
    - automatically matching consumers with relevant advisories
- Preserving security, privacy,
    - no consumer need reveal identity nor attributes
        - no extra risk to consumer if advice not followed
- Offering relevance, timeliness, activity
    - advice typically only visible if relevant to consumer's situation
    - relevance determined automatically by AdviceNet system
    - relevance can include complex combination of conditions:
        - timeliness,
        - hardware attributes,
        - database attributes
        - personal attributes
        - randomization
        - remote attributes

Concrete Instance: Technical Support Industry

AP offers hardware, software, internet service, IT service
AP knows of problematic situations
AP knows precise description of situation preconditions
AP knows precise solution
AP packages information as advisory
AP offers advisory by internet, using Advice Server
AC subscribes
AC has Advice Reader application
Advice Reader reads advisories, determines relevance
Reader reviews, approves, denies
On approval, automatic solution download/install/execute

There are many other concrete instances.

2. Philosophical Overview

Automatic Unattended Operation
infrequent user involvement
periodic unattended synchronizations downloads
constant unattended relevance evaluations

Privacy
One-way nature of subscription interaction
Customer need not reveal information -- name or preferences
Information on client machine stays on the machine

Security
Typically, no advice without subscription
Subscription connotes partial trust
Typically, no effects on system without prior notification and approval
Security through trust
Internet infrastructure allows retraction of one's own faulty advice
Internet infrastructure allows criticism of other people's faulty advice

Decentralization
any IP can be server -- no central registration
any IP can be client -- no special registration

Extensibility
anyone can extend the language to give it new capabilities

3. AdviceNet System Components

AP --	Advice Site	Internet URL-addressible directory on server plurality of files (advisories, inspector libraries) directory message server http/ftp server file server
	advisories	typically specify precondition in formal language typically explain precondition in human language typically explain solution in human language potentially offers internet access to solution to problem specially formatted ascii file easily transported over internet easily created/maintained precise structure in tech report
	site description file	describes an advice site basis for initiating subscription by consumer location (URL) frequency of synchronization type of relationship (free/fee)
	inspector libraries	Special purpose executable code Extends relevance language advice-site specific extension
AC --	advice reader	Application running on client machine Synchronizes with Advice Site Fetches advice files Unwraps advice messages Stores advice messages locally Interprets Relevance Displays Relevant messages to user Manages bodies of advice Manages subscriptions
	inspectors	Invoked by Advice Reader Can inspect properties of machine environment Inspect hardware, file system properties, data in files Allow Relevance decision to involve complex environmental properties.
	user profile	Special user-created file: data on preferences, requirements
	Wizards	support solution process Mr. FixIt Mr. Shell
	advice digests	advice pools site profiles

4. Transaction overview

Subscription model	Users become aware of existence of advice sites Users subscribe
Site Synchronization	Periodically or under user control Advice Reader queries advice site for directory message If directory changed, synchronizes contents Downloads advice from server to client Deletes moot advice on client
Downloading	Uses FTP/HTTP internet mechanism
Interpretation	Advisory: potentially complex hierarchical structure Reader unpacks the components of structure
Relevance Evaluation	Uses formal Relevance Language Parses Relevance Clauses Evaluates Clauses
Inspectors	Evaluate certain phrases in language Called by Advice Reader @ Relevance Evaluation Obtains system information File Properties System Settings File contents Hardware Properties User profile Remote File/System Properties Randomness
Digital Authentication	digital signature on advisory digital authentication of advisory contents
Display	List of relevant advisories Organizational and management tools
Action	User learns something User forwards information to other User on-line participation in timely event User offline action in real world User modify system/database setting Download/install/execute Solution User runs Script File for solution User responds to advice provider by mail,phone, e-mail User purchases object by e-commerce or by phone

5. **Priveleged Sites** Advice Reader typically subscribes to three privileged sites

`Advisories.com`
- Distribution of Advice Reader
- Distribution of Subscription Information

`BetterAdviceBureau.org`
- Edited/Moderated
- Issue advisories against bad advisories
- Issue warnings against bad advice sites
- Compile advice site complaints

`UrgentAdvice.net`
- Issue urgent advice
- Edited/Moderated
- Priority in synchronization

6. **Variations**

Situational Advice	one-time only advice digest
Mandatory Advice	fully automatic solution -- no user approval
Simulated Conditions	method to pretend attributes which don't currently obtain
User side filtering	method to select among relevant ones
Remote Advice Evaluation	

Alternate Transport mechanism
- Advice by e-mail
- Advice by drag and drop
- Advice by file reference

Alternate Notification
- Via Notify Box in other apps
- Via Desktop/Screen Saver
- Via messaging/pager/phone/fax

Open Bi-directional communications
- Questionnaires
- Solution compliance feedback
- Consumer opinion feedback

Masked Bi-directional communications
- Via Anonymity
- Via Randomized Response

Charge Money (metering & cyber$$)
- to publish advice
- to subscribe
- to synchronize
- to download advice
- to download inspector
- to download solution
- to invoke solution

7. Spinoff Opportunities

Commodity Markets
Safety Warnings
Operational Advisories
Product recalls
Portfolio Management
Relevance Mediated Learning/Documentation
Relevance-Filtered Messaging e-mail /web advertising

Example: Commodity.com

Commodity market through AdviceNet
Offereor submits advisory offering object for sale
Advice Site edits and posts
Subscribers input information about interests to user profile
Relevant objects meet their interests

Example: BalanceTransfer.com

BalanceTransfer Offers through AdviceNet
Offeror submits advisory offering balance transfer to sufficient balances
Advice Site edits and posts
Advice Reader uses remote connection to verify balance, preserving privacy
Relevant offers have credit pre-approved

Example: BadPills.com

FDA and other orgs submit info about interactions of medications
Site edits and posts
Pharmacy subscribes
Relevance engine queries database inspector
Database inspector queries SQL database
Relevant Messages = Dangerous Drug combinations

8. Advice File Format

Purpose	to package one or several advisories to offer one or several variants of same advice.
Basic Message:	Wrapper Subject Line Relevance Clause Message Body Action Button
Hierarchical	Advice file = digest of one or more messages Each component potentially relevance-guarded
MIME	Implementation of Wrapper Uses internet standards to package group of messages RFC 822 and successors
Message Body	HTML text both text and HTML
Authentication	Each component of message can be signed Each component of message can be encrypted Digital integrity -- MD5 or similar Digital signature -- PGP or similar

9. Relevance Language

Purpose	To specify precisely conditions under which a certain message would be relevant Ability to describe system state, including hardware, files, file contents, network and remote system states Ability to refer someday to objects in system and world not yet known/created
Characteristics	Descriptive Language English-like Object-Oriented, Strongly Typed Not a procedural language
Limitations	Despite visual resemblance, not like programming language No "`If`" "`While`" "`Case`" "`Goto`" "`Switch`" No traditional variables in language Computed arguments not allowed in some contexts
Security	Evaluation must halt -- No infinite loops No ability to change -- no effects to file system

Extensibility Mechanism	Language can be expanded by adding inspector libraries Defines new data structures Defines new behaviors of those data structures

10. Inspector Libraries

Object-oriented structure	Specify object type Specify all allowed properties Specify outcome type of all allowed properties
Base library	Platform Specific
Extension libraries	Add new components dynamically
Functional Component	Need for extension Need for security
Intellectual Component	Soul of Machine

Special Inspectors

Registry Inspector	find any property of registry
Database Inspector	find any property of general SQL database
User Profile Inspector	find any property of user's profile
Remote Inspector	find any property of other machine in special trust relationship

11. Important System Components Worth Protection

Overall System
Advice Site
Better Advice Bureau Site
Urgent Advice Site
Advice Reader
Inspector libraries
Relevance Language
Document types
Optimizations
Registry Inspectors
Database Inspectors
Questionnaires
Randomized response
Mandatory Solution
Compliance reporting
Remote Relevance Invocation
Wizards: Mr Shell, Mr. Fixit
Automatic advice generation: Naildown

12. Important Abstract Principles Worth Protection

Relevance-Guarded Messaging
Advice-Communication Process
Fourth Branch of Internet
Internet query language/process to remotely determine state of machine
Intellectual Roadmap of Registry
Intellectual Roadmap of OS Resources
Central Nervous System of Computer

Privacy protection while evaluating detailed relevance based on sensitive user information
Internet query language/process to remotely determine sensitive user information
Remote Relevance Querying
Situational Advice

Relevance-Mediated Reading

13. Prior Art

13.1 Common sense

Readme files
Updates via Web Browser
Updates via FTP

13.2 Competitors

Cybermedia
Systemsoft

13.3 Related Ideas: Agents

Firefly
OpenSesame

13.4 Patents

User Profile/Customization of data to consumer
Warning/Notification Systems
Relevance Ranking
Remote Procedure Calls
Process Monitoring Systems

14. Claims

About Relevance-Guarded Messaging

Ability for authors of broadcast messages to specify precisely the conditions under which a message would be of interest to consumer, by describing detailed conditions knowable only within user environment, perhaps after extensive computation. Does not require breach of consumer privacy/security

About AdviceNet System

Automatically connect a population of consumers to *relevant* advisories.

Automatically furnish solutions to those consumers having relevant advisories.

Relevance based on detailed knowledge of information and resources at consumer's site or proxies.

System maintains privacy relationship: although it inspects sensitive information on consumer's site, the advice provider learns nothing about that site through the inspection process

About Priveleged Advice Sites

`Advisories.com, BetterAdviceBureau.com, UrgentAdvice.com`
Mechanism to maintain integrity of an advisory system
Mechanism to identify released advisories and deprecate them

About Inspectors

Mechanism for authors to write in language which can remotely inspect properties of hardware, files, system settings, user profiles, file contents, database contents

Mechanism for authors to extend such language through provision of libraries dynamically extending language syntax, data structures, and semantics

Applications outside of AdviceNet System

About Registry/Database Inspector

Extension of AdviceNet mechanism to know properties recorded in any Microsoft Windows '95/'98 registry

Extension of AdviceNet mechanism to know properties recorded in any standard SQL database.

Applications outside of AdviceNet System

About Remote Inspectors

Extension of AdviceNet mechanism to obtain information about user's environment to data repositories that may lie outside the user's jurisdiction/control. Allows to maintain privacy of distributed sensitive data.

Applications outside of AdviceNet System

About Questionnaires

Mechanism for Data Seeker to enlist consumer populations with special characteristics in data gathering, obtaining automatically detailed information about consumer's situation while offering full consumer control of what is being communicated back to Data Seeker.

About Randomized Response

Mechanism for Data Seeker to enlist consumer populations with special characteristics in data gathering, obtaining automatically detailed information about consumer's situation while actually obtaining no information about any individual consumer.

UCI Website

Web Pages ~~Creation~~
Download Products
Subscription Dictionary
Commercial Transaction Infrastructure

UCI Advice Sites

Better Advice
Urgent Advice
Meta Advice A
UCI Product Advice Site
Incoming User Comments
Advice Site Registration
Advertising Infrastructure

Incoming Messages

If you have Photoshop you should subscribe to Adobe Photoshop' & you are not SUBSCRIBED TO: ______

or you gathered after: ______

Author Side

Installer
Digest Packager
Automated Advice Writers
Automated Solutions
Syntax Checkers
Sample Advice
SDK for Inspector Writers
Documentation
Testing
Internet Commerce Tools
Security/Digital Authentication Tools

Site Description file Creater

Server Side

CGI-Bin Directory Reader
Sample Advice Site
Internet Commerce Tools
Security/Digital Authentication Tools

Client Side

(as listed)
Standard User Profile Database
Standard User Profile Inspectors
Internet Commerce Tools

IT Client

CONFIDENTIAL